VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2016 Tomáš Wagner

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Tomáš Wagner**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: proHR leaders s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Mgr. Ing. Michal Krumnikl, Ph.D.**

Konzultant bakalářské práce: Ing. Lubomír Urbánek

Datum zadání: 01.09.2015

Datum odevzdání: 29.04.2016

doc. Dr. Ing. Eduard Sojka	prof. RNDr. Václav Snášel, CSc.
vedoucí katedry	*děkan fakulty*

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky 1D16 OSTRAVA

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29. dubna 2016 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

proHR
proHR leaders s.r.o.
Nádražní 344/23, 150 00 Praha 5
DIČ: CZ26968584 www.phrl.cz

V Ostravě 29. dubna 2016 .....

Rád bych na tomto místě poděkoval Ing. Lubomíru Urbánkovi za možnost absolvovat individuální odbornou praxi ve společnosti proHR leaders s.r.o. Dále celému týmu, který se na vývoji aplikace proHR podílel, především svému mentorovi Ing. Stanislavu Volnému za ochotu a předané zkušenosti v průběhu naší spolupráce i poté. V neposlední řadě bych rád poděkoval zaměstnancům VŠB za odborné rady při vývoji aplikace.

Abstrakt

Cílem této bakalářské práce je popsat průběh individuální odborné praxe, kterou jsem absolvoval ve společnosti proHR leaders s.r.o. Společnost se zabývá analýzou pracovních procesů a vývojem webové aplikace nazvané "proHR", podporující jejich zdokonalování. Náplní odborné praxe byl vývoj a programování nových i stávajících modulů do aplikace, řešení požadavků klientských společností využívajících tuto aplikaci, a posléze návrh a vývoj administrátorské aplikace pro řízení klientských přístupů.

Klíčová slova: proHR leaders, proHR, PHP, Zend Framework, Symfony, SQL, JSON, AJAX, JavaScript, jQuery, Bootstrap, HTML, CSS

Abstract

The aim of this bachelor thesis is to describe the process of individual professional practice, which I attended in the company named proHR leaders s.r.o. The company focuses on the analysis of work processes and development of web application named "proHR", supporting their improvement. The scope of professional practice was the development and programming of new and existing application modules, addressing the requirements of client companies using this application, and then design and development of the administration application to manage client access.

Key Words: proHR leaders, proHR, PHP, Zend Framework, Symfony, SQL, JSON, AJAX, JavaScript, jQuery, Bootstrap, HTML, CSS

Obsah

Seznam použitých zkratek a symbolů

PHP	–	PHP: Hypertext Preprocessor
SQL	–	Structured Query Language
JSON	–	JavaScript Object Notation
HTML	–	HyperText Markup Language
CSS	–	Cascading Style Sheets
VPN	–	Virtual Private Network
OOP	–	Objektově orientované programování
MVC	–	Model-View-Controller
AJAX	–	Asynchronous JavaScript and XML
XML	–	Extensible Markup Language

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Absolvování individuální odborné praxe jsem si zvolil hlavně proto, že již mám dlouhodobé zkušenosti s prací ve společnosti, která se specializuje na programování a vývoj webových aplikací a stránek. Je to také možnost, jak získat další cenné znalosti a praktické zkušenosti do budoucna.

1.1 Představení společnosti

Společnost proHR leaders s.r.o. se zabývá analýzou pracovních procesů a rozvojem lidských zdrojů. Nabízí aplikaci nazvanou "proHR"pro jednoduché a přehledné monitorování výkonu zaměstnanců v daných oblastech pracovního procesu. Aplikace na základě dat vyhodnocuje pracovní výkon zaměstnanců, jejich silné a slabé stránky, a navrhuje možnosti pro zlepšení výkonu zaměstnanců v daných oblastech.

1.2 Pracovní zařazení

Společnost proHR leaders s.r.o. se však nezabývá vývojem této aplikace po programátorské stránce, pouze navrhuje nové moduly, funkce, vylepšení a úpravy. Programování a vývoj aplikace probíhá pod záštitou partnerské společnosti, která se specializuje na programování a vývoj webových aplikací a stránek, optimalizaci výkonu aplikací a procesní analýzu, a která s proHR leaders s.r.o. velmi úzce spolupracuje. V této společnosti pracuji již delší dobu jako programátor na několika projektech. Jedná se jak o programování a vývoj webových aplikací, tak i o tvorbu webových stránek na základě zadání a požadavků klientů. Pracuji většinou z domu přes VPN připojení, několikrát do týdne docházím na firmu kvůli konzultacím. Tímto způsobem probíhalo také absolvování odborné praxe.

Obrázek 1: Logo společnosti a aplikace

1.3 Představení aplikace

Aplikace proHR je webová aplikace navržená pro společnosti libovolné velikosti. Slouží ke snadnému a přehlednému monitorování a hodnocení pracovního výkonů týmů a zaměstnanců. Na základě hodnocení odhaluje slabá místa zaměstnanců a navrhuje dostupné možnosti, jak se v daných oblastech zlepšit - např. rozvojové aktivity, školení, atd.

1.3.1 Použité technologie

V aplikaci proHR jsou použity aktuální technologie a vývojové metody. V následujícím textu popíšu ty nejdůležitější.

- **PHP** - populární skriptovací jazyk, obzvláště vhodný pro vývoj webových aplikací[1]

  Většina zdrojových kódů aplikace je napsána v PHP verzi 5.6. PHP používáme ve všech webových aplikacích a máme s ním bohaté zkušenosti, proto je použito také v aplikaci proHR. Když jsem se na aplikaci začal podílet, byl již základ aplikace postaven na PHP verze 5.6. Aplikace běží na linuxových serverech, PHP je tedy optimální volba. Za běhu nevyužívá příliš systémových zdrojů, je tedy poměrně rychlé, podporuje **OOP**, má velkou programátorskou komunitu a velké množství frameworků.[2]

- **Zend Framework** - open source PHP framework pro vývoj webových aplikací a služeb[3]

  V aplikaci není použito jen čisté PHP, ale také Zend Framework verze 1.12. Tento framework používáme ve většině webových aplikací, v aplikaci proHR je také použit už od počátku. U verze 1.12 zůstáváme především kvůli zachování syntaxe a struktury aplikace. Framework využívá softwarovou architekturu **MVC**.

- **MySQL** - světově nejpopulárnější open source databáze[4]

  Jako databázový systém je použito MySQL, prozatím nebylo nutné investovat do Microsoft SQL Serveru nebo Oraclu. Jako úložiště dat je použito **MyISAM**.

- **JavaScript** - programovací jazyk HTML a webu[5]

  JavaScript v aplikaci zajišťuje nespočet funkcionalit, především dynamický design, dynamické formuláře a **AJAX**ové požadavky. Použita je také knihovna jQuery.

- **HTML** - značkovací jazyk používaný pro tvorbu webových stránek[6]

  Pro zobrazování obsahu je použito HTML verze 5.

- **CSS** - jazyk popisující způsob zobrazení HTML nebo XML dokumentů[7]

  K úpravě grafické podoby aplikace a stylů je použito kromě JavaScriptu také klasické CSS verze 3. V aplikaci je navíc použita grafická šablona KingAdmin - Responsive Admin Dashboard, která je postavena na **Bootstrap**u verze 3.2.0. Aplikace tak dostala odlehčený, uživatelsky přívětivý, responzivní design - funguje také na na mobilních telefonech a tabletech, na jakékoliv platformě.

1.3.2 Klíčové moduly

Aplikace proHR se skládá z několika modulů. V následujícím textu stručně popíšu nejdůležitější moduly, které zajišťují hlavní funkcionalitu aplikace a konzistenci dat, se kterými aplikace pracuje.

- **competency** - Modul kompetenčních modelů, které se skládají z kompetencí. Každý zaměstnanec musí mít přiřazen právě jeden kompetenční model.
  - příklad: Josef Novák má přiřazen kompetenční model Obchodní ředitel, pod tento model spadají kompetence: Uzavírání smluv, Znalost regionu, Komunikace se zákazníky, atd.
- **people** - Nejrozsáhlejší modul, obsahuje vše, co se týká osob v systému a jejich submoduly.
  - **evaluation** - Modul pro hodnocení výkonu zaměstnance v jeho kompetencích.
  - **goal** - Modul pro zadávání cílů zaměstnanci a sledování plnění těchto cílů.
  - **plan** - Modul pro plánování hodnocení zaměstnance.
  - **team** - Modul pro hodnocení výkonu podřízeného týmu jako celku.
  - **user** - Modul pro práci s informacemi o jednom konkrétním zaměstnanci.
- **plans** - Modul rozvojových plánů, které se skládají z rozvojových aktivit.
  - **activity** - Rozvojová aktivita je článek rozvojového plánu, může rozvíjet jednu nebo více kompetencí.
  - **plan** - Rozvojový plán se skládá z rozvojových aktivit, váže se na konkrétní kompetenční model a může zahrnovat pouze rozvojové aktivity rozvíjející ty kompetence, ze nichž se tento kompetenční model skládá.

Další moduly, které aplikace využívá, jsou modul pro přihlašování, modul pro generování reportů, modul nápovědy, atd.

2 Aplikace proHR

Následující část práce bude pojednávat o řešení mých úkolů v aplikaci proHR. Kromě menších požadavků jsem v této aplikaci řešil tři rozsáhlé úkoly, u každého z nich uvedu zadání, zvolené řešení a jeho implementaci.

2.1 Implementace jazykových verzí

V následujícím textu popisuji zadání a řešení implementace jazykových verzí do aplikace proHR. Možných řešení tohoto úkolu bylo více, v textu jsou popsána dvě, mezi kterými jsem se nakonec rozhodoval.

2.1.1 Zadání

Zadáním mého prvního úkolu v aplikaci proHR bylo implementovat jazykové verze, tedy zajistit možnost, aby se aplikace dala bez problému přepnout do kteréhokoliv jazyka, jehož překlad bude k dispozici. Aplikace byla doposud na úrovni zdrojových kódů napsána v češtině.

2.1.2 Řešení

Z řady možných řešení jsem uvažoval o těchto dvou:

- Databázové tabulky, obsahující dvojce identifikátorů a řetězců, na které se má identifikátor přeložit. Buď společná tabulka pro všechny jazyky, nebo pro každý jazyk jedna tabulka. Dále samozřejmě funkce s identifikátorem jako parametrem, která by se připojila k databázi, podle identifikátoru a požadovaného jazyka vybrala správný řetězec s překladem a vrátila jeho hodnotu do skriptu. Požadovaný jazyk by byl uložen např. v proměnné $_SESSION['Language'], jejíž hodnota by se získávala $_POSTem z nějakého formuláře (select box s jazyky).

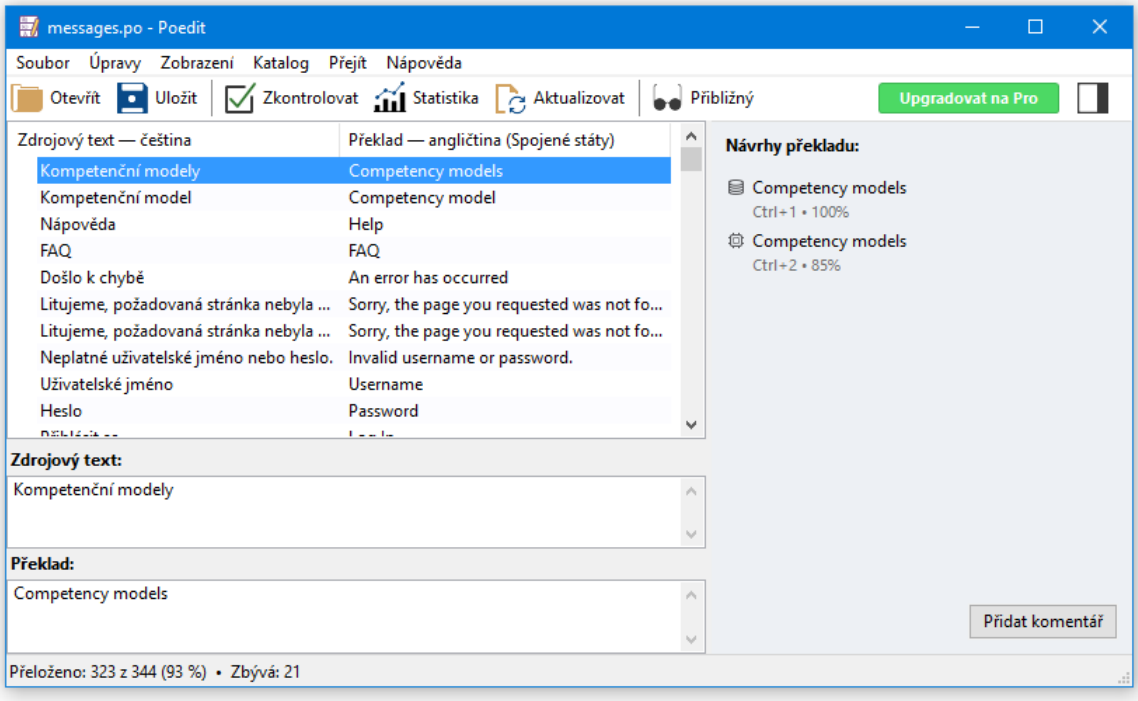
- Řešení pomocí **.po** a **.mo** souborů. Toto řešení se mi zdálo vhodnější, už jen kvůli vyloučení další práce s databází a připojování k ní kvůli každému řetězci. Každý jazyk má svou dvojici .po a .mo souborů. V .po souboru jsou uloženy překlady ve formátu:

  #: [cesta_ke_scriptu]:[cislo_radku] [msgid] ["identifikator"][msgstr] ["prelozeny_retezec"]

  např.:

  #: modules/login/controllers/IndexController.php:169 msgid "Neplatné uživatelské jméno nebo heslo."msgstr "Invalid username or password."

  .mo soubory jsou binární zkompilované .po soubory. Tyto soubory jsou dobře čitelné a modifikovatelné v programu **Poedit**, který jsem za tímto účelem používal.



Obrázek 2: Rozhraní programu Poedit

Jediná potřebná věc pro bezproblémový běh programu je správně nastavit kódování a klíčová slova - jedná se o názvy funkcí, které jsou při překladu volány. Poedit při inicializaci sám prohledá související skripty a zaznamená všechny výskyty těchto klíčových slov. Automaticky se pak zaznamenávají výskyty PHP funkce *gettext()* a její alias *_()*.

Kromě souborů s překlady jsem samozřejmě potřeboval naprogramovat také skripty, které budou práci se soubory řídit. Především skript, který reaguje na změnu jazyka, nastaví proměnné a správné cesty k souborům s překlady. Podporované jazyky jsou uloženy v poli, pokud přijde požadavek na změnu jazyka, ověří se, že je jazyk podporován, uloží se do $_SESSION, nastaví se potřebné *Zend_Registry* a PHP proměnné a nastaví se cesta k souborům s překlady.

V inicializačních skriptech aplikace bylo potřeba nastavit instanci třídy *Zend_Translate* pro překlady. Poslední věc byla vytvořit a graficky přizpůsobit formulář, kterým se budou jazyky přepínat.

Několik problémů nastalo při aplikování překladů, ne vždy šla funkce pro překlad zavolat, protože v některých skriptech nebyla známá. Řešením bylo vyvolání překladače ze *Zend_Registry*. Problém nastal také v případech, kdy texty generoval nějaký JavaScript. V takových případech se musely dané JavaScriptové kódy přepsat, aby jim bylo možné předat již přeložené texty. Některé zdrojové texty, např. pro drobečkovou navigaci, jsou zapsány ve strukturovaných XML souborech. Takovéto soubory se musely pro každý jazyk vytvořit nové.

Aplikace je prozatím přeložena do angličtiny, němčiny a francouzštiny.

```
//podporovane jazyky
$supported_languages = array('cs_CZ', 'de_DE', 'en_US', 'fr_FR');

//mam jazyk a je podporovany
if (isset($_POST["lang"]) AND in_array($_POST["lang"],
$supported_languages)) {
   $language = $_POST["lang"];
}
else if (isset($_SESSION["lang"]) AND in_array($_SESSION["lang"],
$supported_languages)) {
   $language = $_SESSION["lang"];
}
else {
   //nemam jazyk, nastavim defaultni z configu
   $config = Zend_Registry::get('config');
   $language = $config->resources->locale->default;
}

$_SESSION["Language"] = $language;
Zend_Registry::set('Zend_Locale', $language);
$translate = Zend_Registry::get('translate');
$translate->setLocale($language);
putenv("LANG=".$language);
setlocale(LC_ALL, $language);
$folder = APPLICATION_PATH."/../public/langs";
$domain = "messages";
$encoding = "UTF-8";
bindtextdomain($domain, $folder);
bind_textdomain_codeset($domain, $encoding);
textdomain($domain);
```

Výpis 1: Skript pro práci s překlady

2.2 Rozšíření modulu pro cíle

V následujícím textu popisuji zadání a řešení rozšíření modulu pro cíle v aplikaci proHR. Při řešení toho úkolu jsem rozšířil použitý systém zadávání a hodnocení cílů.

2.2.1 Zadání

Jedná se o modul, který zaměstnancům umožňuje zadávat svým podřízeným cíle. Cíle mají termín, do kdy se musejí splnit, a mohou být typu splněno/nesplněno (např. zdokonalit angličtinu). Mým úkolem bylo doprogramovat možnost, kdy mohou být cíle dány také počtem jednotek, kterého se musí dosáhnout (např. uzavřít 25 smluv). Nadřízený má možnost zadané cíle editovat a hodnotit. Pokud je cíl typu splněno/nesplněno, může cíl pouze označit za splněný nebo nesplněný. Pokud je cíl dán počtem jednotek, může pokrok v plnění cíle navíc kdykoliv zaznamenat zadáním aktuálního počtu dosažených jednotek. U cíle se pak objeví progress bar.

2.2.2 Řešení

Databáze

Cílů se v databázi týkají dvě tabulky, *goal* - obsahuje základní informace jako termín splnění, status, vazbu na osobu, atd. a *goal_detail* - obsahuje informace o cíli samotném jako popis cíle, poznámku, informace o plnění cíle a vazbu na cíl.

Do tabulek jsem přidal potřebné atributy, především *date_start* - datum začátku cíle, *number_plan* - plánovaný počet jednotek a *number_real* - dosažený počet jednotek.

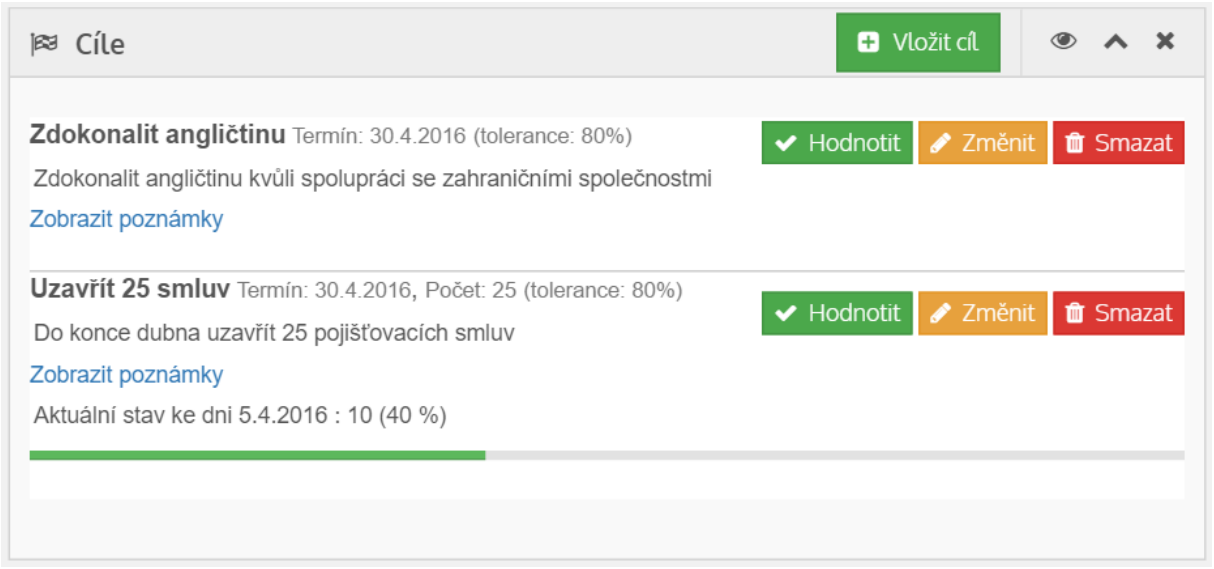
Model

Model pro cíle obsahuje především funkce pro práci s databází. Bylo potřeba upravit funkce tak, aby nové atributy správně vypisovaly a opačně také zapisovaly do databáze.

Controller

V controlleru pro cíle bylo opět potřeba upravit funkce (v Zendu se jim říka *action*) pro zadávání a editaci cílu, kvůli správnému zápisu dat do databáze. U funkce pro editaci jsem musel také upravit data zobrazovaná do view pro formuláře.

Pro možnost zaznamenávání pokroku bylo potřeba naprogramovat řídící funkci. Funkce reaguje na kliknutí na tlačítko pro hodnocení pokroku cíle. Důležitým parametrem pro funkci je identifikační číslo cíle, který se má hodnotit. Podle tohoto parametru se pomocí funkcí z modelu zjistí detail cíle a osoby. Proběhne ověření, zda má přihlášená osoba práva tento cíl hodnotit, a pomocí JavaScriptu se vyvolá okno s požadovaným layoutem. Pokud je cíl typu splněno/nesplněno, zobrazí se dvě tlačítka, jedním se cíl označí jako splněný, druhým jako neplněný, a v obou případech se cíl deaktivuje, odebere se zaměstnanci z aktivních cílů a přesune se do historie cílů. Pokud je cíl dán počtem jednotek, zobrazí se třetí tlačítko a pole, kam se zadá aktuální počet dosažených jednotek. Tlačítkem se pak provede kontrola vstupů, aktuální počet dosažených jednotek se uloží, cíl zůstává aktivní. V seznamu aktivních cílů se u něj poté objeví progress bar.



Obrázek 3: Aktivní cíle z pohledu nadřízeného

View

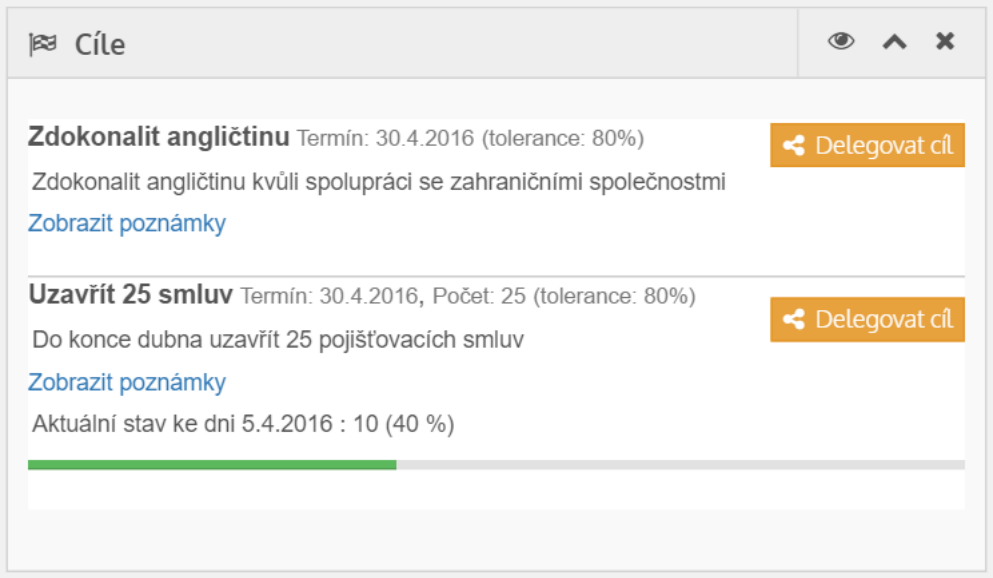
Ve view jsem provedl pouze změny ve formulářích pro zadávání a editaci cílů, na tlačítko pro hodnocení pokroku cíle jsem nastavil vyvolání okna pomocí JavaScriptu, a nastavil atributy pro řízení zobrazování progress barů. Progress bary jsou již součástí grafické šablony KingAdmin - Responsive Admin Dashboard, stačilo se podívat na demo a správně je zobrazit.

2.2.3 Delegování cílů

Později přišel požadavek na doprogramování funkcionality, kdy zaměstnanec, který má od nadřízeného zadaný cíl, může tento cíl delegovat na své podřízené. Zaměstnancům, kteří mají podřízené a zadaný cíl, přibude tlačítko pro delegování jednotlivých cílů na podřízené.

V controlleru bylo potřeba naprogramovat novou funkci, která bude na stisknutí tlačítka reagovat. Funkce pracuje se dvěma parametry - identifikačním čísle osoby, která deleguje cíl na podřízené a identifikačním číslem delegovaného cíle. Formulář pro delegování cíle je podobný jako pro zadávání, s tím rozdílem, že některé hodnoty jsou již vyplněny, dají se však upravit. Předvyplněné informace o delegovaném cílí se získají pomocí funkcí z modelu pro cíle podle druhého parametru - identifikační číslo delegovaného cíle. Jediná nutnost je vybrat osoby ze seznamu podřízených, kterým se cíl zadá. Tento seznam podřízených se získá pomocí funkcí z modelu pro osoby podle prvního parametru - identifikačního čísla osoby. Po odeslání formuláře proběhne kontrola vstupů a každému zvolenému podřízenému se cíl zapíše jako aktivní.

Toto řešení je prozatím v testovací fázi, v požadavku nebylo uvedeno, co se má stát s původním delegovaným cílem, ani zda mají být delegované cíle nějakým způsobem propojeny. Očekávám, že dojde k upřesnění zadání, a tato funkcionalita se bude ještě dále upravovat.



Obrázek 4: Aktivní cíle z pohledu zaměstnance

2.3 Implementace modulu rozvojových plánů a aktivit

V následujícím textu popisuji zadání a řešení implementace rozvojových plánů a aktivit do aplikace proHR. Návrh řešení tohoto úkolu byl zcela v mé režii, postup řešení jsem konzultoval se svým mentorem a následně s nadřízeným.

2.3.1 Zadání

Nejrozsáhlejším zadáním, které jsem v průběhu odborné praxe dostal, bylo návrh, naprogramování a implementace nového modulu pro rozvojové plány a aktivity.

Základem je rozvojová aktivita. Jedná se o školení nebo podobnou událost, která rozvíjí jednu nebo více kompetencí. Může rozvíjet kompetence spadající pod různé kompetenční modely.

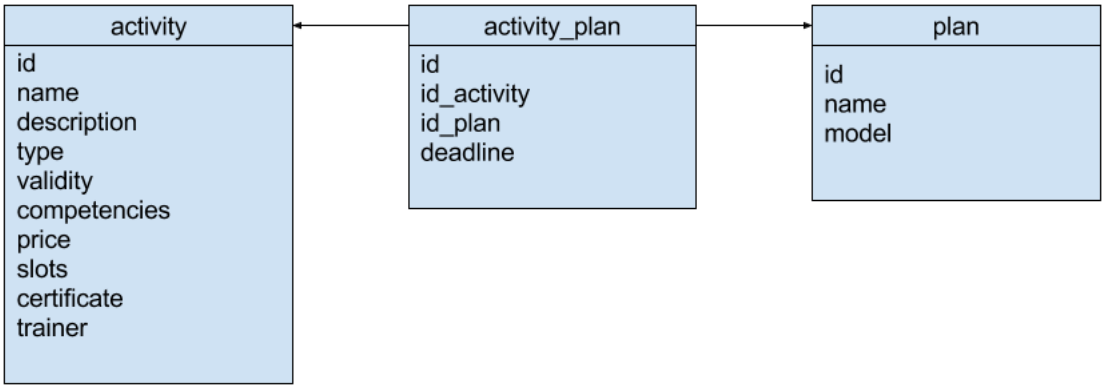
Rozvojový plán je soubor rozvojových aktivit, které rozvíjejí alespoň jednu aktivitu z kompetenčního modelu, na který je rozvojový plán navázán. Stručně řečeno, rozvojové aktivity rozvíjejí kompetence, rozvojové plány rozvíjejí celé kompetenční modely.

Dále bylo potřeba naprogramovat mechanismus, který podle hodnocení zaměstnanců v jejich kompetencích vybere ty zaměstnance, jejichž úroveň hodnocení je nižší, než je požadovaná úroveň, a doporučí tyto zaměstnance pro absolvování dané rozvojové aktivity.

2.3.2 Řešení

Databáze

1. *activity* - atributy rozvojové aktivity, napojení na tabulku *competency* pomocí pole identifikačních čísel kompetencí
2. *plan* - atributy rozvojového plánu, napojení na tabulku *competency_model* pomocí identifikačního čísla kompetenčního modelu
3. *activity_plan* - vazební tabulka pro propojení rozvojových aktivit a plánů



Obrázek 5: Databázové tabulky pro rozvojové aktivity a plány

V databázi jsem vytvořil výše uvedené tabulky. Atribut *competencies* v tabule *activity* slouží jako pole identifikačních čísel kompetencí, do databáze se ukládají jako řetězec čísel, oddělených čárkami (např. *3,10,22*). Řetězec se ve skriptech rozparsuje a s identifikačními čísly se dále pracuje. Při zápisu do databáze se naopak identifikační čísla, oddělená čárkami, zapíší do řetězce a ten se poté zapíše do databáze. Využil jsem funkcí *explode()* a *implode()*.

Model

Pro každou přidanou tabulku jsem naprogramoval model pro práci s databází. V každém modelu se nacházejí standardní funkce pro výpis všech řádků z tabulky, výpis jednoho řádku z tabulky podle zadané podmínky, zápis dat do tabulky, editace dat v tabulce a mazání dat z tabulky. Model pro tabulku *plan* obsahuje navíc funkci pro získání posledního vloženého záznamu.

```php
foreach (explode(',', $activity['competencies']) AS $id_competency) {
   // info o kompetenci
   $competency = $mCompetency->getCompetency($id_competency);
   // ...dalsi prace s kompetencemi
}
```

Výpis 2: Použití funkce explode()

```php
$data4DbActivityInsert = array (
   'competencies' => isset($formData['competencies']) ?
   implode(",", $formData['competencies']) : "",
   // ...dalsi atributy pro zapis do databaze
);
```

Výpis 3: Použití funkce implode()

Controller

Controllery jsem naprogramoval dva, jeden pro plány a jeden pro aktivity.

Controller pro rozvojové aktivity se skládá z funkcí pro přidávání, editaci, mazání a výpis aktivit.

Funkce pro přidávání a editaci jsou dost podobné, proběhne kontrola vstupů, a pomocí funkcí z modelu se zapíší data. Funkce pro editaci navíc podle parametru, kterým je identifikační číslo aktivity, načítá data o aktivitě z databáze a předvyplňuje jimi editační formulář.

Funkce pro mazání pouze ověří parametr, kterým je opět identifikační číslo aktivity a pomocí funkce z modelu smaže příslušný záznam aktivity.

Funkce pro výpis aktivit jsou dvě, jedna pro výpis všech aktivit a druhá pro detailní výpis konkrétní aktivity. Funkce pro výpis všech aktivit si pomocí funkce z modelu vytáhne z databáze všechny aktivity a pošle je v poli do view, kde se vysázejí do přehledné tabulky s několika detaily. Funkce pro detailní výpis konkrétní aktivity si podle parametru, kterým je identifikační číslo aktivity, pomocí funkce z modelu vytáhne z databáze požadované údaje o dané aktivit a podle těchto údajů si zjistí další informace pro detailní výpis. Podle rozparsovaného řetězce identifikačních čísel kompetencí si pomocí funkce z příslušného modelu zjistí všechny kompetence, ke kterým se aktivita váže. Ke každé kompetenci si také pomocí funkce z příslušného modelu zjistí kompetenční model.

Dále si pro každého zaměstnance podle jeho posledního hodnocení ověří, zda tento zaměstnanec nemá v dané kompetenci nižší hodnocení, než je požadované hodnocení. Pokud ano, zařadí ho do seznamu zaměstnanců, kteří jsou označeni jako adepti na absolvování této rozvojové aktivity. Nakonec pošle tento seznam do view, kde jsou v detailu každé aktivity tito adepti uvedeni.

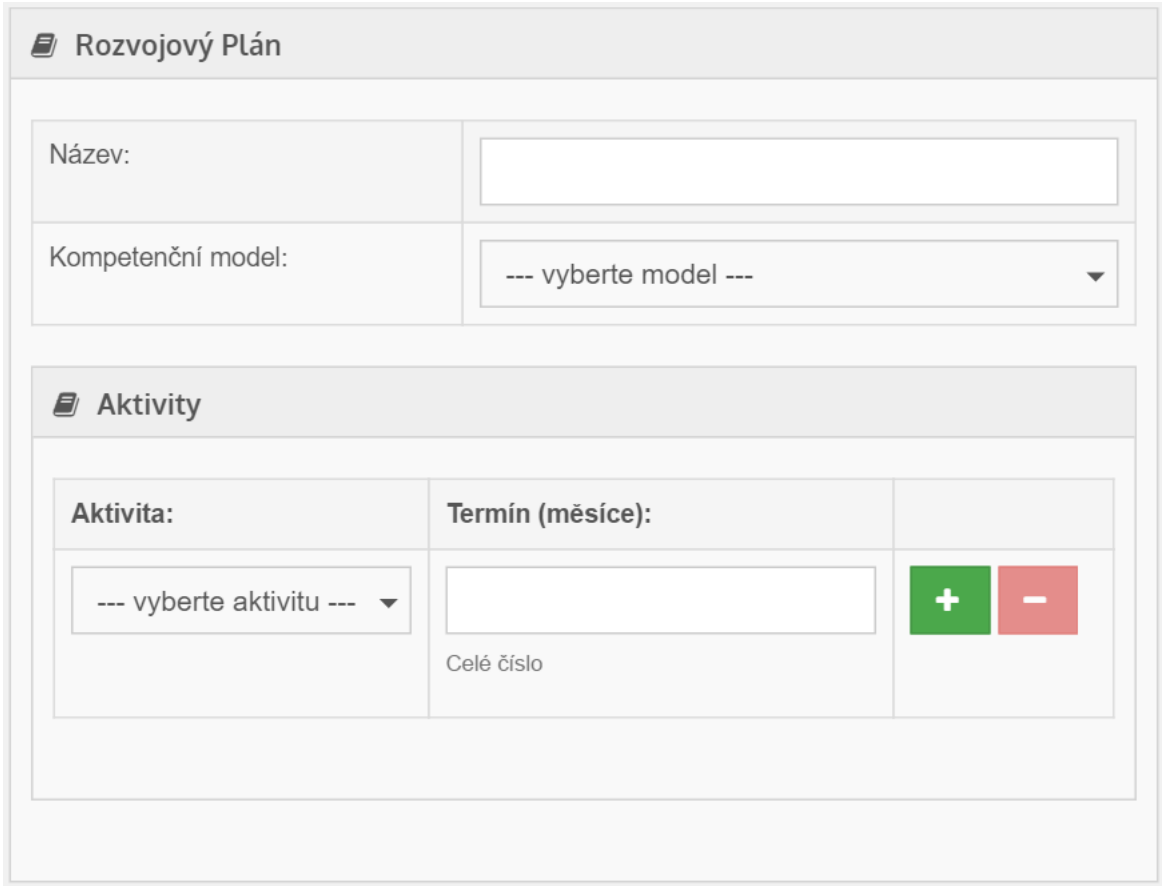
Controller pro rozvojové plány obsahuje stejné funkce jako controller pro aktivity, má však některé funkce navíc.

První funkce představuje možnost okamžitého odebrání aktivity z rozvojového plánu a reaguje na kliknutí na tlačítko, které tuto funkci plní. Tlačítko je umístěno na každém řádku ve výpisu navázaných rozvojových aktivit, který je součástí detailu rozvojového plánu. Funkce odebírá aktivitu na základě dvou parametrů - identifikační číslo aktivity a identifikační číslo plánu. Příslušný záznam je smazán pouze z vazební tabulky *activity_plan*, plán ani aktivita se z databáze nevymažou, pouze vazba mezi nimi.

Druhá funkce umožňuje generování dynamických formulářů pro vkládání rozvojových plánů. Implicitně má zakázáno zobrazovaní výstupu, slouží pouze pro volání AJAXovou funkcí při změnách formuláře.

Funkce je nutná pro dodržení omezení, kdy se rozvojový plán může skládat pouze z aktivit rozvíjejících kompetence z kompetenčního modelu, ke kterému se plán váže. Ve formuláři pro vkládání rozvojového plánu se nejprve vybírá kompetenční model, ke kterému se plán bude vázat. Dále se přidávají aktivity, které bude plán zahrnovat. Aby bylo omezení dodrženo, musí byt ve výběru k dispozici pouze aktivity rozvíjející kompetence z vybraného kompetenčního modelu.

Funkce podle si parametru, kterým je identifikační číslo zvoleného kompetenčního modelu zjistí kompetence, které pod něj spadají. Dále si zjistí všechny dostupné aktivity. Postupně porovná všechny kompetence z modelu s kompetencemi, které aktivity rozvíjejí. Pokud najde shodu, uloží identifikační číslo aktivity a její název do pole, z něhož se pak generuje HTML výstup pro výběr vyhovujících aktivit. Vygenerovaný HTML kód pak odešle AJAXové funkci.



Obrázek 6: Formulář pro vkládání rozvojových plánů

```
public function selectbyAction() {
      $this->_helper->layout()->disableLayout();
      $idModel = $this->_getParam('model', 0);
      $mCompetency = new Competency_Model_Competency();
      $mActivity = new Plans_Model_Activity();
      $competencies = $mCompetency->getCompetency4Model($idModel);
      $activities = $mActivity->getActivities();
      $spolecne = array();

      foreach($activities AS $activity) {
         $competencyIDs = explode(",", $activity['competencies']);
         foreach($competencies AS $competency) {
            if(in_array($competency['id'], $competencyIDs)) {
               if(!in_array([$activity['id'], $activity['name']], $spolecne)) {
                  array_push($spolecne, [$activity['id'], $activity['name']]);
               }
            }
         }
      }

      $vypis = '<option value selected>--- '.$this->view->trans("vyberte
          aktivitu").' ---</option>';

      foreach($spolecne AS $row) {
         $addSelect = '';
         if(isset($_POST['activity']) AND $row[0] == $_POST['activity']) {
            $addSelect .= ' selected="selected"';
         }
         $vypis .= '<option value="'.$row[0].'"'.$addSelect.'>'.$row[1].'</
             option>';
      }

      echo json_encode($vypis, JSON_UNESCAPED_UNICODE);
      exit;
}
```

Výpis 4: Funkce pro výběr vyhovujících aktivit

```
$("#model").on("change", function () {
   $id_model = $('#model').val();
   $.ajax ({
      url: "/plans/plan/selectby/model/" + $id_model,
      dataType: "json",
      type: "GET",
      success: function(data) {
         $(".activity").html(data);
      }
   });
});
```

Výpis 5: Asynchronní funkce reagující na změnu kompetenčního modelu ve formuláři

View

Pro view jsem musel naprogramovat několik skriptů, jak pro aktivity, tak pro plány. Většinou v Zendu platí, že ke každé funkci v controlleru patří jeden script ve view, výjimkou je výše uvedená funkce pro výběr vyhovujících aktivit, kde jsem zobrazování výstupu do view zakázal. Ve funkcích pro přidávání a editování je použit převážně JavaScript, kvůli dodržení požadované dynamiky formulářů. Kromě výše uvedené asynchronní funkce jsem naprogramoval také funkce pro dynamické přidávání a odebírání polí formuláře.

Aktivity se do systému prozatím vkládají mechanicky pomocí formuláře, do budoucna se však uvažovaje o napojení na nějakou databázi obsahující kurzy, školení, atd. V úvahu připadá např. databáze portálu *www.edumenu.cz*, tato záležitost však ještě není dořešena.

Průběžně, dle příležitostných požadavků, jsem také do aplikace implementovat různé doplňky, např. další progress bary a grafy pro znázornění různých výpočtů, nebo jQuery plugin, který na stránce zvýrazňuje zvolené prvky a krátce popisuje jejich funkcionalitu. Slouží tak jako jednoduchá, snadno dostupná nápověda.

3 Klientské subdomény a administrátorská aplikace

Poslední věc, na které jsem v rámci aplikace proHR podílel, bylo navržení a implementace řešení, kdy každá klientská společnost bude mít svou subdoménu, na které bude aplikace provozována, a také navržení a implementace administrátorské aplikace, která bude umožňovat řízení práv a přístupů klientů k těmto aplikacím.

3.1 Klientské subdomény a demoverze

V následujícím textu popisuji zadání a návrh řešení klientských subdomén a demoverzí. Otázku nastavení webového serveru jsem konzultoval s příslušným správcem, otázku správy a nastavení databází se zkušeným databázovým administrátorem.

3.1.1 Zadání

Každá klientská společnost bude mít svou subdoménu, reprezentovanou identifikačním názvem, např. *sodexo.prohr.cz*. Ze začátku může využívat demoverzi, po jejím skončení musí za využívání aplikace zaplatit.

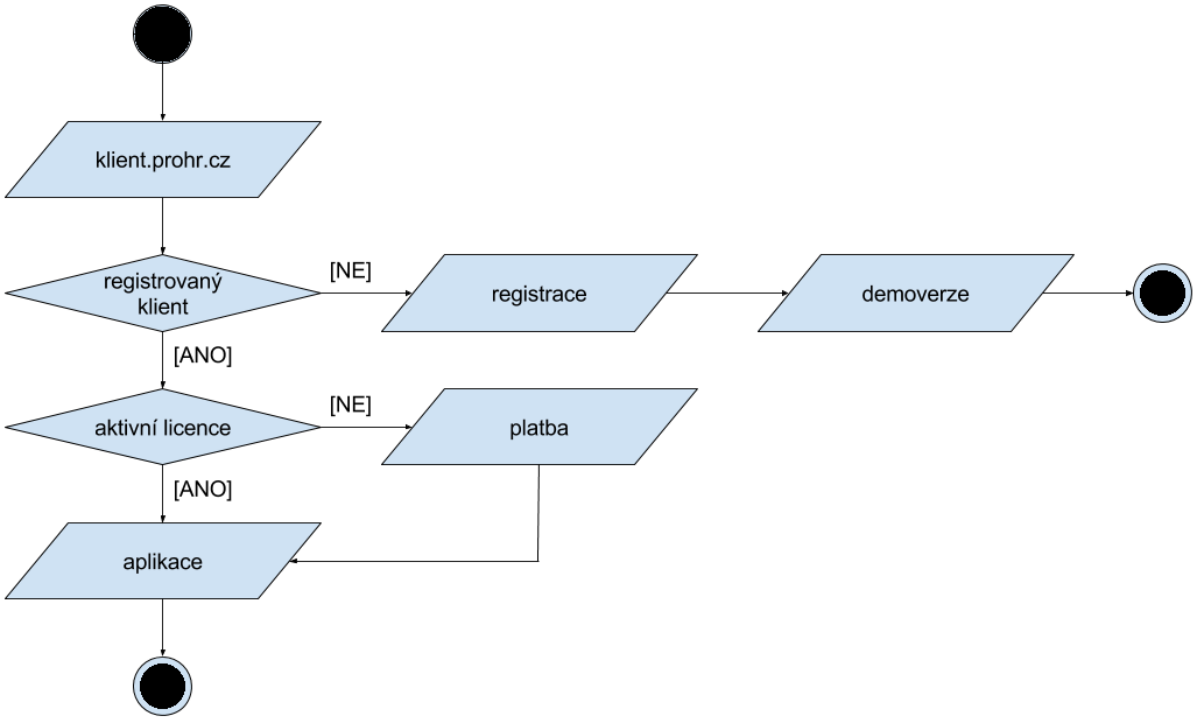
3.1.2 Návrh řešení

Webový server bude nastaven tak, aby přijímal požadavky na jakoukoliv subdoménu **.prohr.cz*. Databáze pro toto řešení bude obsahovat tabulku s identifikačními názvy klientů, statusy klientských aplikací, informacemi o licenci atd.

Prvotní skript ověří, zda je požadovaná subdoména vedena jako identifikační název klienta. Pokud ne, návštěvník je přesměrován na registrační formulář, kde mu bude po vyplnění údajů zpřístupněna demoverze aplikace a vytvořen záznam s identifikačním názvem v tabulce. Dále mu bude vytvořena databáze se stejným názvem. Pokud již klient v databázi veden je, ověří se, zda má aktivní licenci, tedy zda má zaplacen přístup do aplikace. Pokud ne, návštěvník je přesměrován na stránku pro provedení platby. Pokud má aktivní licenci, spustí se aplikace s příslušnou databází.

name	**created**	**demo**	**active**	**employees**	**price_per_employee**
sodexo	2015-09-01	NO	YES	200	10
lenovo	2016-01-01	NO	YES	10 000	5
librex	2016-03-30	YES	NO	NULL	NULL

Tabulka 1: Příklad databázové tabulky pro správu klientů



Obrázek 7: Diagram chování prvotního scriptu

Navržené řešení se v tuto chvíli projednává, otázka některých procesů, jako jsou např. zabezpečení při vytváření nových databází pro klienty nebo řešení plateb není ještě zcela uzavřena. Bude také potřeba zavést nějaký mechanismus pro filtrování rezervovaných a nevhodných názvů subdomén.

3.2 Administrátorská aplikace

V následujícím textu popisuji zadání a návrh řešení administrátorské aplikace. Tuto aplikaci bude využívat především můj nadřízený, její návrh jsem proto konzultoval primárně s ním. Pro její implementaci se mi podařilo prosadit novou verzi PHP a také nový framework.

3.2.1 Zadání

Administrátorská aplikace bude sloužit k řízení práv a přístupů klientů k aplikacím. Administrátor bude moci zřizovat nebo prodlužovat demo přístupy na libovolnou dobu, řídit dostupnost modulů v rámci aplikace, kontrolovat počet zaměstnanců, vedených v aplikaci, generovat a odesílat měsíční faktury za užívání aplikace, atd. V aplikaci bude použito PHP verze 7 a framework Symfony verze 3.

Symfony je open source PHP framework pro vývoj webových aplikací.[8] Kombinaci PHP verze 7 a frameworku Symfony verze 3 jsme zvolili především kvůli zvýšení rychlosti a zajištění plynulého běhu aplikace. Výběr frameworku jsem konzultoval se zkušeným vývojářem webových aplikací.

3.2.2 Návrh řešení

Administrátorská aplikace bude pracovat se stejnou databází, jako prvotní skript, řídící klientské subdomény. Z této databáze budou řízeny demo přístupy a další nastavení klientských aplikací. Pro řízení dostupnosti modulů v aplikacích bude potřeba upravit controllery všech modulů a zavést podmínku, která bude kontrolovat, zda je v aplikaci tento modul zpřístupněn. Nastavení dostupnosti modulů bude pravděpodobně uloženo v databázi aplikace. Z administrátorské aplikace bude možno toto nastavení spravovat.

Kontrolu počtu zaměstnanců, vedených v každé aplikaci, by mohla obstarávat procedura na databázovém serveru. Ze všech databází aplikací by sečetla počet vedených zaměstnanců a uložila by je do tabulky pro správu klientů.

Dříve, než se začne řešit konečná verze tohoto řešení a implementace výsledné aplikace, musí být uzavřena a vyřešena otázka klientských subdomén. Prozatím jsem vytvořil nový virtuální server, nainstaloval na něj webový server, PHP verze 7 a framework Symfony verze 3, abych si nastudoval a vyzkoušel syntaxi, používanou v tomto frameworku, strukturu aplikace, atd.

4 Závěr

Přestože webové aplikace a stránky programuji již delší dobu, práce na aplikaci proHR pro mě bylo částečně něco nového.

Poprvé jsem pro implementaci jazykových verzí použil .po a .mo soubory a naučil se užívat programy a postupy s tím spojené. Toto řešení se mi zdá lepší a vhodnější než překládání řetězců z databáze a budu ho používat nadále v dalších projektech.

Poprvé jsem také naprogramoval složitější funkci, která pouze odesílá data jako JSON objekty, a asynchronní JavaScriptovou funkci, která zmíněnou funkci využívá k získávání dat pro změny hodnot dynamických formulářů bez nutnosti obnovení stránky.

Při návrhu databází a tabulek jsem využil znalosti získané během studia na vysoké škole. Značně jsem si rozšířil dovednosti a znalosti programovaní v JavaScriptu, jejichž základ jsem také získal během studia.

Nadále budu pokračovat ve zdokonalování dovedností a znalostí frameworku Symfony verze 3. Ve společnosti momentálně dále pracuji na několika dalších projektech, řeším a vyřizuji tikety na helpdesku a podílím se na vývoji několika webových aplikací.

Současná verze aplikace proHR je již v provozu, momentálně ji využívá několik desítek spokojených klientských společností, jako nejznámější bych rád uvedl např. společnost Sodexo s.r.o.

Reference

[1] *PHP: Hypertext Preprocessor*. URL: http://php.net.

[2] *10 Advantages of PHP over other languages*. URL: http://www.webnethosting.net/10-advantages-of-php-over-other-languages.

[3] *About - Zend Framework*. URL: http://framework.zend.com/about.

[4] *MySQL :: About MySQL*. URL: http://www.mysql.com/about.

[5] *JavaScript Tutorial*. URL: http://www.w3schools.com/js.

[6] *HyperText Markup Language – Wikipedie*. URL: http://cs.wikipedia.org/wiki/HyperText_Markup_Language.

[7] *CSS Tutorial*. URL: http://www.w3schools.com/css.

[8] *About Symfony Project*. URL: http://symfony.com/about.